Comparison of DES and A New Cryptosystem

A Thesis Submitted

in Partial Fulfillment of the Requirements

for the Degree of

Master of Technology

by

PRATIMA GUPTA

to the

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

May 1998

CSE-1998-M-GUP-COM

CERTIFICATE

This is to certify that the work contained in the thesis entitled "**Comparison of DES and A New Cryptosystem**" by **Pratima Gupta** has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

[illegible]

Dr. Manindra Agrawal,
Assistant Professor,
Department of Computer Science & Engineering,
Indian Institute of Technology, Kanpur.

Abstract

Cryptography is the art of secret writing, and cryptanalysis is the art of breaking ciphers. Data Encryption Standard (DES) is one of the most popular iterative cryptoscheme, but it is very much sensitive to differential cryptanalysis. In this thesis, a new cryptosystem is analysed to compare it with DES. First it is compared on the basis of the resistivity to differential cryptanalysis and then to observe its resistivity to other cryptanalysis schemes, we proceed to implement several randomness tests and apply these tests on both cryptosystems (DES and the new cryptosystem). We also implement an automated password generator using DES which produces random yet pronounceable passwords. Our studies reveal that the new cryptosystem is superior to the DES against differential cryptanalysis scheme and, for small number of rounds, produces more random output, which makes it more difficult to analyse with respect to other cryptanalysis schemes.

Acknowledgement

I take this opportunity to express my heartfelt gratitute to my thesis supervisor **Dr. Manindra Agrawal** for having introduced me to this topic. But for his encouragement, support and patience, this thesis would not have seen the light of the day. I remain indebted to Dr. Harish Karnick for his useful suggestions at the beginning of this study.

During the course of my study here, I had the privilege of attending enlightening lectures on cryptography by Dr. M. U. Siddiqui.

The encouragement from my parents and bhaiya-bhabhi coupled with their confidence in my abilities motivated me to pursue higher studies at this institute. The regular visits by my brother Sachin helped me overcome homesickness. My special thanks to Manish for inspiring me to strive hard for professional excellence.

Finally I would like to thank my friends Manoj, Vishal, Shailesh, Rajesh, Sunita and all of my batchmates, whose joyful company I am going to miss. Without their timely assistance this work could not have been completed.

Contents

List of Figures

Chapter 1

Introduction

In the early days, when organisations had a single computer center, it was very easy to achieve security by keeping disks, tapes or cards secure from unauthorized users. But with the advent of networking, the situation has changed completely. No one can guard the data, containing millions of bits that daily move between the computers. Organisations have no way of being sure that their data are not copied by the wiretap, or other means on the way to their proper destination. And now in the modern era, when transmission is done on the satellite links, data may be available to anyone with the help of an antenna. And that is the motivation for using cryptography, the art to make data unintelligible to all, but the intended recipient.

The uses of cryptography are typically the authentication of the source of a given message, the authentication of data itself in the message, and the guaranteeing of the secrecy of the message. Thus it plays an important role specially in military, commercial or business applications where lots of confidential material has to be transmitted.

1.1 Cryptography and Cryptanalysis

Cryptography - the science of secrecy - is the process of scrambling data to hide its content so that valuable or sensitive information can be protected from unauthorized disclosure. Its purpose is to ensure security and privacy by keeping the information hidden from anyone for whom it is not intended. Encryption and decryption are the key features of cryptography.

The basic idea of cryptography is as follows. The message to be encrypted is known as *plaintext* P, which is transformed by a encryption function E to convert it into an unreadable form. The cryptoscheme uses some *key* K in order to convert P into this unreadable form, called *ciphertext* T.Thus

$$T = E\left(K, P\right)$$

There is also a general decryption function D in which the key K and a ciphertext T may be put in order to produce message P.

$$P = D\left(K, T\right)$$

Any such system has the following properties:

(i) Decryption reverses encryption :

$$D\left(K, E\left(K, P\right)\right) = P$$

(ii) It is "impractical" to decrypt ciphertext, without the appropriate key.

Clearly, to maintain the security of such a system, the key must be kept secret. The goal of the cryptographer is to make decryption as difficult as possible for unintended users(i.e. those without knowledge of key).

However, as desirable as it is to keep one's information private, it may also be desirable for someone to break through this encryption to read a message even if one is not the intended receiver. Breaking a scheme involves determining what the plaintext P was from T, without prior knowledge of the key K. It may involve finding out what the key K used was, or it may involve creating an inverse operation that for a given cryptoscheme, will reliably produce P from T without knowledge of K.

In any cryptosystem, keeping security by not letting anyone know the details of cryptoscheme is not considered a good method in general, though it does make the practical task of breaking cryptoscheme more difficult. Therefore, the cryptanalyst is always assumed to be familiar with the cryptoscheme that they are attempting to break. However unlike the intended recipient he does not know what the key is and so cannot decrypt the cipher text easily. The art of breaking ciphers is called *Cryptanalysis*. The art of devising ciphers (cryptography) and breaking them (cryptanalysis) is collectively known as *Cryptology*.

Cryptography is a contest between two adversaries: the designer of the system and the opponent, who attempts to recover the plaintext and/or key, and breaks the cryptoscheme.

1.2 Motivation

Data Encryption Standard (DES) is one of the most popular private key (there is only one key that is kept secret) iterative cryptosystem. There are many reasons for using DES. First of all, it is very easy to implement in hardware. It is based on permutations and rotations of a vector of bits and these operations are quite easy to do in hardware. Secondly cryptanalysis of DES is not a trivial task. In fact, no method was known to break DES faster than a brute force search until 1990. In 1990 differential cryptanalysis was introduced and it was found that DES is very sensitive to this attack, because of some of its inherent properties. Motivation of this thesis work is to analyse a new cryptosystem to compare its resistivity with that of DES with respect to different attacks. Firstly we consider differential cryptanalysis based attacks and show that the new cryptosystem is far less sensitive to such attacks. Next, we consider the resistivity of the two systems against any kind of attack. We take the approach of measuring the randomness in the ciphertext produced by two systems. If the ciohertext is completely random then clearly no attack can succeed.

Towards this end, we carry out several randomness tests on the ciphertext. It is observed that the new cryptosystem (with 3 rounds) produces output that is more random than DES (with 16 rounds). Curiously, the output of the new cryptosystem with 10 rounds is far less random than that of 3 rounds! We try to give some reasons for this phenomenon.

We also implement an automated password generator using DES as a random number generator so that password created by this generator are completely random and pronounceable.

1.3 Organisation of the Report

The report is organised in the following way:

Data Encryption Standard (DES), and the way to differential cryptanalyse DES is presented in *Chapter 2*. An example to break three round DES is also illustrated.

In *Chapter 3* a new cryptosystem is analysed against differential cryptanalysis. Its resistivity is compared to the DES's resistivity to differential attack. This chapter briefly discusses various randomness tests which are useful for our applications.

Results of differential cryptanalysis on three round DES are discussed in *chapter 4*. Randomness tests are applied on both (DES and new cryptosystem) and a comparison is done on the basis of these tests too. This chapter also discusses these results, and the future scope of the thesis work.

Chapter 5 explains the implementation of automated password generator using DES.

Chapter 2

DES Cryptosystem and Its Differential Cryptanalysis

2.1 Iterative Cryptosystems

Iterative cryptosystems are a family of cryptographically strong functions based on iterating a weaker function (the function, which is not very difficult to break) n times. Each iteration is a *round* and the cryptosystem is called an n round cryptosystem. Each round is a function of the output of the previous round and subkey is key dependent value, calculated via key scheduling algorithm. $S - boxes$ are non-linear translation tables mapping a small number of inputs to a small number of outputs.

Probably the best known algorithm of this type is the Data Encryption Standard (DES). Because DES is widely used, it is the focus of the research on the strength of iterative cryptosystems and, for the same reason, this is used as the example in this thesis work.

2.2 Data Encryption Standard (DES)

This scheme is introduced in 1970, and in 1977 National Bureau of Standards announced it as the official US standards for sensitive but un-classified information.

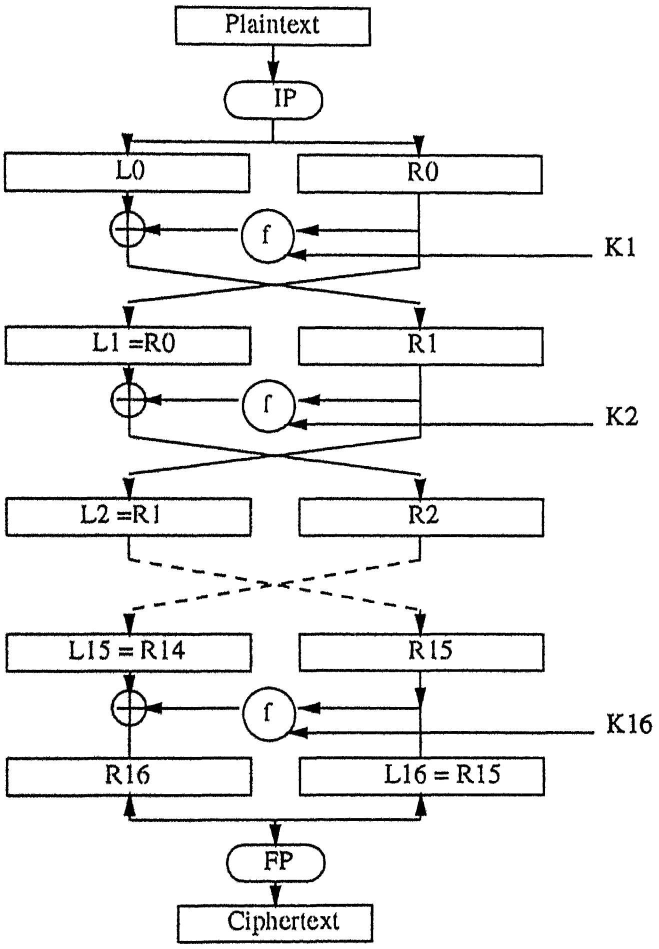

Figure 1: DES

DES [2] works on a block of 64 bits and encrypts each block at a time. The encrypted block is also 64 bits long. The way it is done is that the block is permuted through *Initial Permutation (IP)* and then it is split into the two halves. The right half enters the *F function,* or *F box,* which also takes as input a subkey, and then the result of the F box is XORed with the left half. Then the new left half becomes the right half, while the old right half (before going into the F function), becomes the new left half. That process is called a *round.* Full DES calls for 16 rounds. The subkey is a subset of the key (a vector with 56 bits of information, and 8 checksum or parity bits). The particular subset used is determined by permutation and rotation.

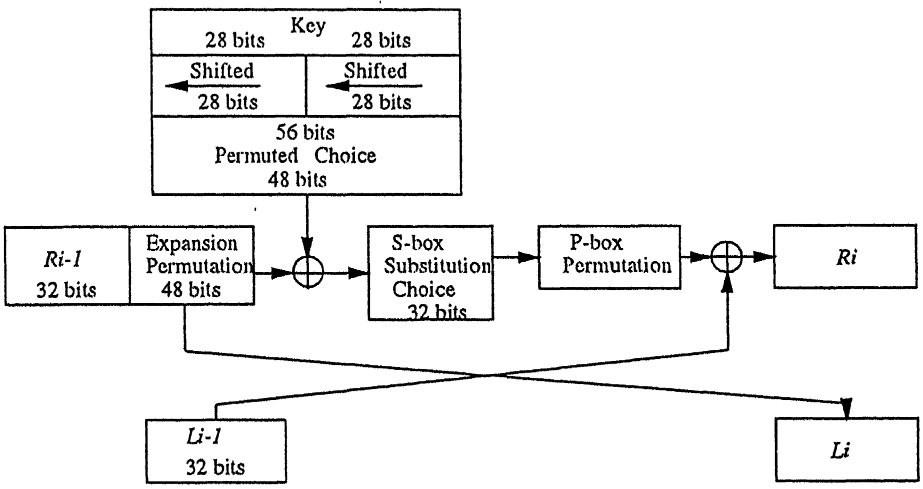

Figure 2: One round of DES

As the diagram (Fig. 2) shows, the key is split into halves (28 bits per half, the checksum bits are ignored), and each half is rotated left. The halves are then sent as the input to another permutation function which splits out the 48 bit subkey. The result of the rotation however is used to determine the next subkey (is rotated again, and again). Because the key halves are rotated again and again, a different subkey is used in each round, increasing the security of DES. After the last round, the data is permuted through *Final Permutation (FP)* and the output is final encrypted block.

The F function takes a 32 bit vector as input, as well as a 48 bit subkey, and has as output 32 bits. The input is expanded through an *Expansion permutation E* which repeats certain bits of the vector in a fixed manner to turn the input into a 48 bit vector. This 48 bit expanded vector is XORed with the subkey k (which is also 48 bits). The resultant of the XOR is sent through *S boxes* which translate 6 bit vectors into 4 bit vectors. The first 6 bits of the 48 bit vector would go through S1, the next 6 to S2, and so on, up to S8. The values of the first and last bit of the six bit vector are used to determine the line to read for the S box. The middle 4 bits determine which column to read. So, if input to the S1 is 101110, then according to 0 based numbering it would look up 3rd line and 8th column of S1. The other S boxes work in a similar manner, but with different look up tables. The output of the eight S boxes (a 32 bit vector) is sent through a permutation *P Box Permutation,* which merely twists the bits around in some set fashion.

Decryption of DES is done by feeding T into DES backwards. Since all of the operations are reversible, decryption is essentially as fast as encryption.

2.3 Differential Cryptanalysis of DES

In 1990, Eli Biham and Adi Shamir introduced the notion of differential cryptanalysis. This is a chosen plaintext attack against DES and more efficient than brute force. This scheme works on many pairs of plaintext, with some particular difference, and then difference in the resultant ciphertext pair is analysed for each plaintext pair.

If the XOR value of a plaintext pair is taken, then it remains invariant in the XORing of the key and is linear in the *E expansion*, the *P permutation* and *XOR operation* between the left half of the data and the output of the function. But DES also contains nonlinear *S−boxes* and in differential cryptanalysis only S boxes are focussed.

Several definitions and concepts are given below to justify the reasons of focussing on the S boxes. In these functions, X is some plaintext or input, X^* is some other plaintext or input, and $X \oplus X^*$ is denoted as the difference between X and X^* or the input XOR (or plaintext XOR).

The F function uses an expansion function, this function relates the output pair difference to the input pair difference when expansion permutation is applied.

$$E(X) \oplus E(X^*) = E(X \oplus X^*) \qquad (1)$$

Also the subkey is XORed with the output of the E function. This function relates two different inputs to the XOR function that are being XORed with the plaintext. This formula is given by:

$$(X \oplus K) \oplus (X^* \oplus K) = (X \oplus X^*) \tag{2}$$

Further, the F box also has a P permutation. The formula which relates input pair XOR and output pair XOR into the P permutation is given as:

$$P(X) \oplus P(X^*) = P(X \oplus X^*) \tag{3}$$

Finally, a function to relate the XOR function that connects the different rounds is needed. This function is:

$$(X \oplus Y) \oplus (X^* \oplus Y^*) = (X \oplus X^*) \oplus (Y \oplus Y^*) \tag{4}$$

Y and Y^* are some pair of inputs just like X and X^*, but for the other input coming into the XOR function. The reason for these formulas is so that we may focus on the most important and most controversial part of the DES cryptoscheme, the S boxes. The S boxes are made controversial by the arbitrariness in how they were chosen. It is S boxes that provide the security to DES. As the above formulas show, the rest of the DES function is not overly hard to analyse. Thus the main part of the cryptanalyser is to analyse the S boxes.

As indicated above, in S boxes the vector of length 6 determines some output that is a number from 0 to 15 (which can be expressed as a vector of length 4). It means knowledge of input XOR can not guarantee the knowledge of output XOR.

Input	Output XOR															
XOR	0_x	1_x	2_x	3_x	4_x	5_x	6_x	7_x	8_x	9_x	A_x	B_x	C_x	D_x	E_x	F_x
0_x	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1_x	0	0	0	6	0	2	4	4	0	10	12	4	10	6	2	4
2_x	0	0	0	8	0	4	4	4	0	6	8	6	12	6	4	2
3_x	14	4	2	2	10	6	4	2	6	4	4	0	2	2	2	0
4_x	0	0	0	6	0	10	10	6	0	4	6	4	2	8	6	2
5_x	4	8	6	2	2	4	4	2	0	4	4	0	12	2	4	6
6_x	0	4	2	4	8	2	6	2	8	4	4	2	4	2	0	12
7_x	2	4	10	4	0	4	8	4	2	4	8	2	2	2	4	4
								⋮								
38_x	0	6	2	2	2	0	2	2	4	6	4	4	4	6	10	10
39_x	6	2	2	4	12	6	4	8	4	0	2	4	2	4	4	0
$3A_x$	6	4	6	4	6	8	0	6	2	2	6	2	2	6	4	0
$3B_x$	2	6	4	0	0	2	4	6	4	6	8	6	4	4	6	2
$3C_x$	0	10	4	0	12	0	4	2	6	0	4	12	4	4	2	0
$3D_x$	0	8	6	2	2	6	0	8	4	4	0	4	0	12	4	4
$3E_x$	4	8	2	2	2	4	4	14	4	2	0	2	0	8	4	4
$3F_x$	4	8	4	2	4	0	2	4	4	2	4	8	8	6	2	2

Table 2.1: Partial pairs XOR distribution table of S1 (values subscripted with x are in hexadecimal)

Differential cryptanalysis begins by making what is called a difference distribution table for each S box (table 2.1). This table is a chart of input XORs and output XORs, and the possible pairs exist with that status. If we denote some input XOR as X, and some output XOR as Y, we say that X may cause Y by the S box if there is some pair of inputs, for which the *input* $XOR = X$ and the *output* $XOR = Y$ for that particular S box. The average entry is 4. If however, there is no such pair of inputs to the S box, for which the *input* $XOR = X$ and the *output* $XOR = Y$, we say that X may not cause Y by the S box.

This table can be used as a tool to identify key bits. If the output XOR of the F function of the last round is known, and pair of ciphertext is known then the bits of possible subkeys can be filtered out. Using both ciphertexts it is easy to calculate the input XOR of the F function of the last round and its output XOR, then the input XOR and output XOR of each S box in the last round are known. If there are k input pairs, when referred to table, then exactly k values of corresponding six subkey bits are possible.

As an example, given that the input XOR and the output XOR is 0 (the inputs are identical and the outputs are identical) there are 64 possible such pairs. This result should make sense since in this case, $X = X^*$ (they have no difference), so it makes sense that there are 64 possible such pairs (one for each possible input to S1). Similarly, we should note that if the input XOR is 0 (the two inputs are identical), all other output XORs have no possible pairs. (The outputs must be identical in that case.)

Here an introductory example to break three round DES is explained [4].

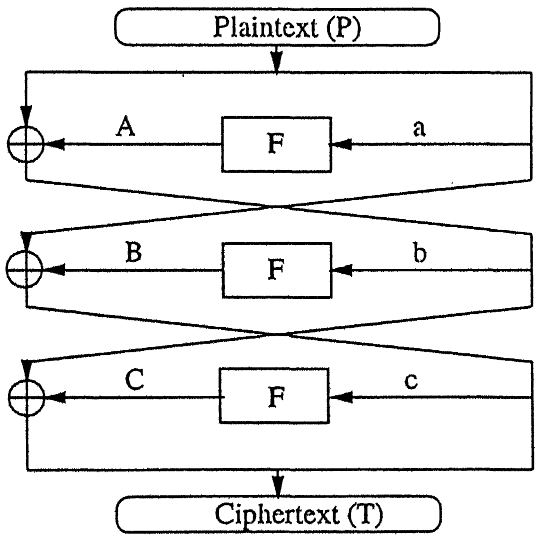

Figure 3: DES with three rounds

This attack analyses the pairs of ciphertexts whose corresponding plaintext XOR's consist of a zero right half and arbitrary left half. The main part of the algorithm is to find out pairs which suggest each possible value entering each S box. For each ciphertext pairi, the input XOR of the function F in the first round is zero and thus its output XOR must be zero.The left half of the ciphertext is calculated as the XOR value of the left half of the plaintext, the output of the first round and the output of the third round ($l = L \oplus A \oplus C$, where L is the left half of the plaintext and l is the left half of the ciphertext, See Fig. 3). Now, the plaintext XOR is fixed, the ciphertext XOR is known and the output XOR of the first round is zero. Therefore, the output XOR of the F function in the third round can be calculated as the XOR of the left half of the plaintext XOR and left half of the ciphertext XOR. The inputs of the F function in the third round can be obtained easily from the ciphertext pair.

S box input	Possible keys
06, 32	07, 33
10, 24	11, 25
16, 22	17, 23
1C, 28	1D, 29

Table 2.2: Possible keys for 34, D by S1 with input 1, 35 in hexadecimal.

The following analysis is done for each S box in the third round, here only S1 is taken to show the analysis. The input XOR and output XOR of S1 can be easily extracted with the help of input XOR and output XOR of the F function in the third round. Suppose S_E is the value of input bits which are XORed with the six key bit(denoted by S_K) to make input to the S box (denoted by S_I), and S_I can be obtained from ciphertext pair.

If k is the number of possible input pairs to S1 corresponding to that input XOR and output XOR, then exactly k key values are suggested by the pair via $S_K = S_E \oplus S_I$. For example, if the ciphertext bits which are input to the third round S1 are $S_E = 1$ and $S_E^* = 35$. The output XOR (known by ciphertext pair XOR) is D then the value of S_K must appear in table 2.2. The row of the table shows two pairs with the same two inputs but with two possible orderings. Each pair extracts one possible value of the key, so from each row two possible values of key can be extracted (which are $S_E \oplus S_I$ and $S_E^* \oplus S_I$).) Now, this experiment is repeated for many pairs, The possible values of key bits entering in S1 are noted down, and the value which is suggested by all the pairs must be the real value of key bits.

This differential scheme is implemented for three round DES, and the results are discussed in chapter 4.

For n round cryptosystems the basic tool, which helps us to push the knowledge of the plaintext XOR to a knowledge of an intermediate XOR, is called *n round characteristics.* Every n round characteristics has a particular plaintext XOR Ω_P, a particular XOR of the data in n^{th} round Ω_T, and a probability p^{Ω} when random pairs whose plaintext XOR is Ω_P are used. By using these characteristics one can find out the information about the key bits, but the amount of information one obtains is dependent on how many rounds are used for DES. For a many round version of DES, less information can be obtained, making analysis more difficult. For reduced round variations of DES, Biham and Shamir claim that their method can break DES in under two minutes for 8 round variations, and far faster for fewer rounds. From these results, it becomes worthwhile to study their method, and the cryptographer is once again reminded that one cannot rely on a scheme to remain secure.

Chapter 3

A New Cryptosystem and Its Comparison With DES

The main purpose to introduce differential cryptanalysis was to analyse the security of DES- like cryptosystems, and it was found that because of some design deficiencies these cryptosystems are very sensitive to differential cryptanalysis. So, researchers try to design new cryptosystems which are unaffected by the differential cryptanalysis. Some new design criteria are proposed, which are claimed to provide immunity to differential cryptanalysis, and here we analyse such a new cryptosystem with its new properties.

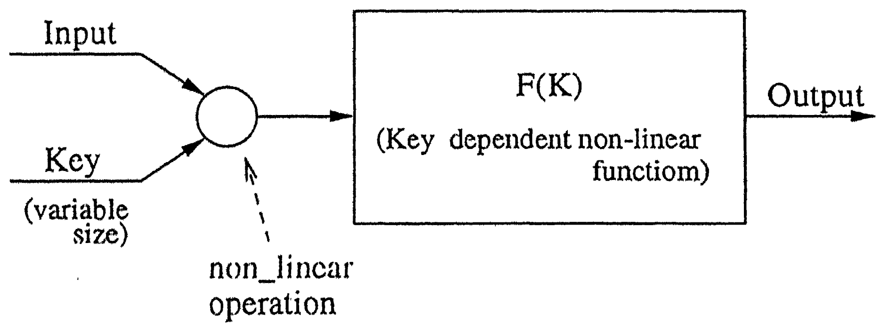

Figure 4: Non-linear functionality of new cryptosystem

The structure of new cryptosystem [1]is almost similar to DES, except the F function of new cryptosystem contains non-linear function F as S-boxes in DES with some added properties.

1. Key interact non-linearly with plaintext.
2. Key governs the non-linear transformations.
3. This function F allows for arbitrary key size.

We can compare this cryptosystem to DES on the basis of *functionality* and by applying *randomness tests* on the output sequences. The first section compares the functionality of the two cryptosystems in view of differential cryptanalysis, and second section gives a brief description of the randomness tests applied on ciphertexts. Results of the randomness tests are discussed in chapter 4.

3.1 Functionality

The main requirement for the differential cryptanalysis is to find out good characteristics, i.e. to find out pairs of plaintext (with particular differences), such that the difference of the outputs of n^{th} round is predictable with a relatively high probability.

The first property of new cryptosystem says that the key operates non-linearly on input. Remember that in case of DES, input was XORed (linear operation) with key and therefore XOR of the actual inputs to non-linear function (S-boxes) was same as the XOR of the inputs (which were operated with key bits to get actual inputs). In other words, we can say that in DES the characteristics were key independent and there was a single distribution table for each of the S box. But in new cryptosystem there may be a different distribution table for each of the key value, and it means that the characteristics are key dependent.

If the characteristics are key dependent, then their probability is defined as the probability that a random pair is right pair (whose plaintext difference is Ω_P), with

[1]This new cryptosystem has been given to us by Indian Navy for analyis, and its details are confidential. Hence, we are not able to describe it completely.

respect to a random key. However, the probability that a random pair is a right pair with respect to a fixed key may depend on the choice of the key. So for these type of *conditional characteristics*, probability that a random pair is right pair vary for different keys. In other words, this property of cryptosystem reduces the probability of characteristics.

The second property of the cryptosystem is "key governs the non-linear transformation". In DES, output was fixed for a given value of input to non-linear function (S boxes), but in this case the transformation will depend on the key value. Since the non-linear function is key dependent, it becomes quite difficult to find characteristics. Especially the characteristics which can predict the output of all the keys, have a very low probability. Even after applying first property, if still there are some characteristics which have high probability for all the keys or for a large fraction of key space, the second property of the cryptosystem makes the differential cryptanalysis infeasible.

The third property of the new cryptosystem that "function F allows for arbitrary key size" shows that key space is increased as compared to DES. In DES, resistivity to differential cryptanalysis was not dependent on the key space because neither the key interacted non-linearly with plaintext, nor non-linear function was key dependent. But in the new cryptosystem, having key dependent characteristics, increased key space makes differential cryptanalysis almost impossible.

So these properties, when applied together, prove that this is a good attempt by the designer to strengthen his algorithm against differential cryptanalysis.

3.2 Randomness Tests

Our main purpose is to check the random behaviour of output sequences. The theory of statistics provides us with some quantitative measures for randomness. There may be a large number of tests that can be conceived; but here only those tests are performed which have proven to be most useful, most instructive, and most readily adaptable to computer calculation. If a sequence behaves randomly with respect to n tests, it does not indicate that the sequence is completely random, yet each test gives us more and more confidence in the randomness of the sequence. In practice we apply about half a dozen different kind of statistical tests to a sequence, and if sequence passes these tests satisfactorily, then we consider it to be random. To test the sequence *empirical tests* are carried out, for which the computer manipulates groups of numbers of sequence and evaluates certain statistics.

3.2.1 Chi-square Test (χ^2 Test)

It is a general test procedure for studying random data and the best known of all statistical tests, and it is a basic method that is used in connection with many empirical tests. In general, n observations are taken in an experiment and every observation can fall into one of k categories. We take n *independent observations*; this means that the outcome of one observation has absolutely no effect on the outcome of others. Let p_s be the probability that each observation falls into category s, and let Y_s be the number of observations that actually do fall into category s. The formula for variance V is

$$V = \sum_{1\leq s\leq k} \frac{(Y_s - np_s)^2}{np_s} \tag{5}$$

By expanding $(Y_s - np_s)^2 = Y_s^2 - 2np_sY_s + n^2p_s^2$ in Eq. 5 and using the facts that

$$Y_1 + Y_2 + \ldots + Y_k = n, \tag{6}$$

$$p_1 + p_2 + \ldots + p_n = 1, \tag{7}$$

we arrive at the formula

$$V = \frac{1}{n}\sum_{1\leq s\leq k}\left(\frac{Y_s^2}{p_s}\right) - n \tag{8}$$

This makes the computation of V easier. Now it is very important to know, what constitutes a reasonable value of V. This is found by referring to chi-square distribution table [1], which gives the values of chi-square distribution with ν "degrees of freedom" for various values of ν; the number of " degrees of freedom" is $k-1$, one less than the number of categories.

The number of observations falling into each of k categories is counted and quantity V given in equation eq. 8 is computed. Then V is compared with the numbers given in the table, with $\nu = k-1$. If V is less than 1% entry or greater than the 99% entry, then the numbers as not sufficiently random. If V lies between 1% and 5% or between 95% and 99% entries the numbers are "suspect"; if V lies between the 5% entries and 10% entries, or the 90% and 95% entries, the number might be "almost suspect." The chi-square test is often done at least three times on different sets of data, and if two of the three results are suspect then the numbers are regarded as not sufficiently random.

3.2.2 Empirical Tests

In this section 9 kinds of specific tests are discussed, that have been applied to sequences in order to investigate their randomness. Here auxiliary sequence

$$\langle Y_n \rangle = Y_0, Y_1, Y_2, \ldots, \tag{9}$$

is used. This is a sequence of integers in which all elements are independent and uniformly distributed between 0 and $d-1$. The value of d should be large enough so that the test is meaningful, but not so large that the test becomes impracticably difficult to carry out.

▪ *Equidistribution Test(Frequency Test)*

The first requirement that sequence must meet is that its numbers are uniformly distributed between 0 and $d-1$. For each integer r, $0 \leq r < d$, count the number of times that $Y_j = r$ for $0 \leq j < n$, and then apply the chi-square test using $k = d$ and probability $P_s = 1/d$ for each category.

Serial Test

The pairs of successive numbers should be uniformly distributed in independent manner. To carry out this test, the number of times that the pair $(Y_{2j}, Y_{2j+1}) = (q, r)$ occurs is counted, for $0 \leq j < n$; these counts are to be made for each pair of integers (q, r) with $0 \leq q, r < d$, and the chi-square test is applied to these $k = d^2$ categories with probability $1/d^2$ in each category.

For more accurate results, we can generalize this test to triples, quadruples, etc., instead of pairs.

Gap Test

This test is to examine the length of "gaps" between occurences of Y_j in a certain range. if α and β are two real numbers with $0 \leq \alpha < \beta \leq d-1$, we want to consider the lengths of consecutive subsequences $Y_j, Y_{j+1}, \ldots, Y_{j+r}$ in which Y_{j+r} lies between α and β but the other $Y's$ do not.

In this algorithm for any given value of α and β number of gaps of *length* $0, 1, \ldots, t-1$ and the numbers of gaps of *length* $\geq t$ is counted until n gaps have been tabulated. If $Count[r]$ for $0 \leq r \leq t-1$ represents the number of gaps of length r and $Count[t]$ is the number of gaps of *length* $\geq t$, then chi square test is applied to the $k = t-1$ values of $Count[0], Count[1], \ldots, Count[t]$, using the following probabilities:

$$p_0 = p, \quad p_1 = p(1-p), \quad p_2 = p(1-p)^2, \quad \ldots, \\ p_{t-1} = p(1-p)^{t-1}, \quad p_t = (1-p)^t. \tag{10}$$

Here $p = (\beta - \alpha)/d$.

When these tests are applied with $(\alpha, \beta) = (0, d/2)$ or $(d/2, d)$ then these tests are called "runs above the mean" and "runs below the mean," respectively.

▪ *Poker Test(Partition Test)*

The "classical" poker test considers 5 successive integers, $(Y_{5j}, Y_{5j+1}, \ldots, Y_{5j+4})$ for $0 \leq j < n$, and counts the number of *distinct* values in set of five. We would then have five categories:

5 different	=	all different;
4 different	=	one pair;
3 different	=	two pair, or three of a kind;
2 different	=	full house, or four of a kind;
1 different	=	all of a kind;

In general we can consider n groups of k successive numbers, we can count the number of $k-$ tuples with r different values. A chi-square test is then made using the probability

$$p_r = \frac{d(d-1)\ldots(d-t+1)}{d^k} \left\{ \begin{matrix} k \\ d \end{matrix} \right\} \tag{11}$$

that there are r different.

▪ *Coupon Collectors Test*

In this test we observe the segments $Y_{j+1}, Y_{j+2}, \ldots, Y_{j+r}$ to collect a complete set of integers from 0 to $d-1$, and this segment is called "coupon collector" segment. This algorithm counts the length of n consecutive "coupon collector" segments. If *Count*$[r]$ is the number of segments with length r, for $d \leq r < t$, and *Count*$[t]$ is the number of segments with *length* $\geq t$, then a chi-square test is applied to *Count*$[d]$, *Count*$[d+1]$, $\ldots$, *Count*$[t]$, with $k = t - d + 1$, where corresponding probabilities are

$$\begin{aligned} p_r &= \frac{d!}{d^r}\left\{ \begin{matrix} r-1 \\ d-1 \end{matrix} \right\}, \quad d \le r < t; \\ p_t &= 1 - \frac{d!}{d^{t-1}}\left\{ \begin{matrix} t-1 \\ d \end{matrix} \right\}; \end{aligned} \tag{12}$$

▮ *Permutation Test*

In this test the input sequence is divided into n groups and each group contains t elements. Now the elements in each group can have $t!$ orderings; the number of times each ordering appears is counted, and a chi-square test is applied with $k = t!$ and with probability $1/t!$ for each ordering. For example, if $t = 3$ there may be 6 categories according to whether $Y_{3j} < Y_{3j+1} < Y_{3j+2}$ or $Y_{3j} < Y_{3j+2} < Y_{3j+1}$ or ... or $Y_{3j+2} < Y_{3j+1} < Y_{3j}$, where $Y's$ in a group should not be equal.

▮ *Run Test*

A sequence may be tested for "runs up" and "runs down." In this test we examine the length of increasing or decreasing segments. If $Count[r]$ is the number of segments with length r, for $d \le r < t$, and $Count[t]$ is the number of segments with $length \ge t$, then a chi square test is applied to $Count[d]$, $Count[d+1]$, ..., $Count[t]$, with $k = t - d + 1$, where corresponding probabilities are

$$p_r = \frac{1}{r!}\left(1 - \frac{1}{r+1}\right), \quad 1 \le r < t; \qquad p_t = \frac{1}{t!} \tag{13}$$

▮ *Collision Test*

The collision test counts the number of collisions, and the sequence generator is assumed random number generator if the collisions occured are not too many or too few.

Suppose there are m urns and n balls are thrown randomly into those urns, If a ball falls into an urn that already contains at least one ball, we say that a "collision" has occured.

Suppose $m = n^{20}$ and $n = 2^{14}$. Then each urn will receive only $64th$ of a ball, on an average. The probability that a given urn will contain exactly k balls is

$$p_k = \binom{n}{k} m^{-k}\left(1 - m^{-1}\right)^{n-k}, \tag{14}$$

so the expected number of collisions per urn is

$$\begin{aligned} \sum_{k\geq 1}(k-1)p_k &= \sum_{k\geq 0} kp_k - \sum_{k\geq 1} p_k \\ &= \frac{n}{m} - 1 + p_0 \end{aligned} \tag{15}$$

Since

$$p_0 = \left(1 - m^{-1}\right)^n = 1 - \frac{n}{m} + \binom{n}{2} m^{-2} + smaller \quad terms, \tag{16}$$

we find that the average number of collisions taken over all m urns is very slightly less than $n^2/2m = 128$.

Serial Correlation Test

The "serial correlation test" is the measure of the amount Y_{j+1} depends on Y_j and it is computed as follows

$$C = \frac{n(Y_0Y_1 + Y_1Y_2 + \ldots + Y_{n-2}Y_{n-1} + Y_{n-1}Y_0) - (Y_0 + Y_1 + \ldots + Y_{n-1})^2}{n\left(Y_0{}^2 + Y_1{}^2 + \ldots + Y_{n-1}{}^2\right) - (Y_0 + Y_1 + \ldots + Y_{n-1})^2} \tag{17}$$

In general, if we have n quantities $Y_0, Y_1, \ldots, Y_{n-1}$ and $Z_0, Z_1, \ldots, Z_{n-1}$, the correlation coefficient between them is defined to be

$$C = \frac{n\sum(Y_jZ_j) - (\sum Y_j)(\sum Z_j)}{\sqrt{\left(n\sum Y_j^2 - (\sum Y_j)^2\right)\left(n\sum Z_j^2 - (\sum Z_j)^2\right)}} \tag{18}$$

where $0 \leq j < n$.

A correlation coefficient always lies between -1 and $+1$. When it is zero or very small, it indicates that the quantities Y_j and Z_j are relatively independent of each other, but when the correlation coefficient is -1 or $+1$, it shows total linear independence.

Instead of computing the correlation coefficient between the observations $(Y_0, Y_1, \ldots, Y_{n-1})$ and their immediate successors $(Y_1, Y_2, \ldots, Y_0)$, we can also compute it between $(Y_0, Y_1, \ldots, Y_{n-1})$ and any cyclically shifted sequence $(Y_q, \ldots, Y_{n-1}, Y_0, \ldots, Y_{q-1})$. the cyclic correlation should be small for $0 < q < n$.

These tests are applied on both (DES and new cryptosystem). Results are discussed in last chapter. (Chapter 4), which show that output sequence of new cryptosystem is atleast as random as the output sequence of DES.

Chapter 4

Results and Discussion

4.1 Results

4.1.1 Differential Cryptanalysis

The differential cryptanalysis is implemented on three round DES (as explained in chapter 2). With the help of this, we can find out 48 bit (out of 56 bits). Rest of the bits can be found out by exhaustive search. The table 4.1 shows the number of plaintexts used and time taken by the system for different key values.

Key in hexadecimal	Number of plaintext pairs used	Time taken by system (sec)
0117 0304 0506 0708	103	0.04
0A14 2005 025A 0301	128	0.04
1815 0309 0543 221F	105	0.04
0A02 425A 0420 1F07	104	0.04
1542 5A04 1F0B 0600	105	0.04

Table 4.1: Software implementation of differential cryptanalysis for three round DES

4.1.2 Randomness Tests

Randomness tests are applied on the ciphertexts of same size obtained from both the cryptosystems. Here in this chapter, to show and discuss the results, we introduce the following abbreviations:

(i) Data Encryption scheme with 16 rounds and 64 bits is denoted by DES;

(ii) New cryptosystem with 3 rounds, key length approx. 900 bits is NCS, and

(iii) New cryptosystem with 10 rounds and key length approx. 14000 bits is represented by LCS.

To apply these results, we took a large number of plaintext files and obtained the corresponding ciphertext files for DES, NCS and LCS.

Then we took a fixed block size and applied all the randomness tests. For frequency test, serial test, gap test, poker test, permutation test and run test, chi-square distribution values are calculated, and graphs show *chi-square value* on the x-axis and *number of observations* on y-axis. These tests are performed for two different block sizes. Selected percentage points (at 1%, 5%, 25%, 50%, 75%, 95%, 99%) of the chi-square distribution are respresented by vertical dashed lines. Results of collision test are shown by the graph, by taking *number of collisions* on the x-axis and *number of observations* on y-axis for a fixed block size. For correlation test, *correlation coefficients* are taken on x-axis and *number of observations* are taken on y-axis.

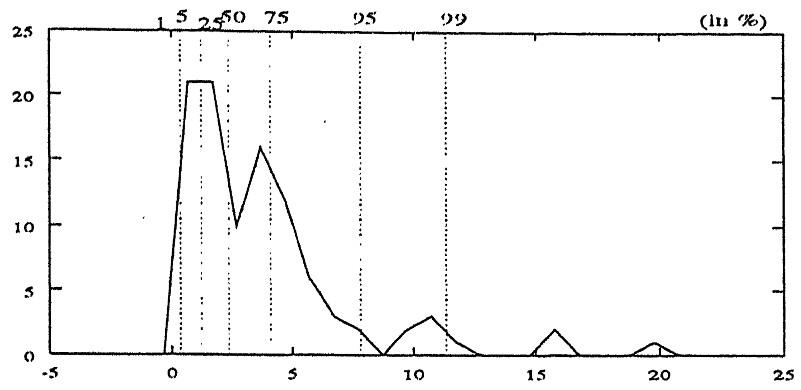

Figure 5: Frequency Test (for DES). Block size =2 bits.

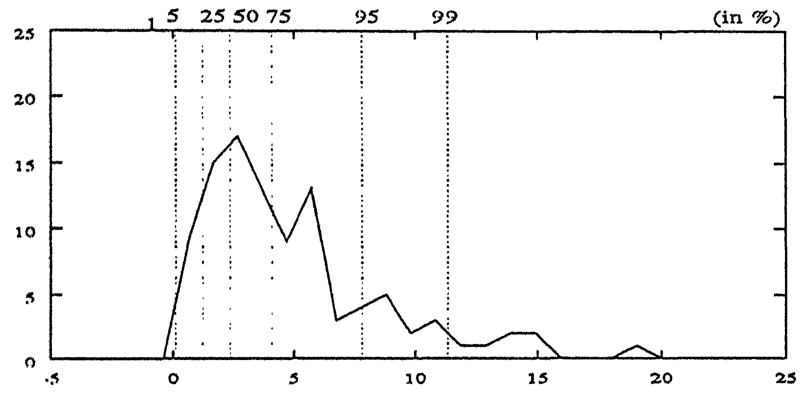

Figure 6: Frequency Test (for NCS). Block size =2 bits.

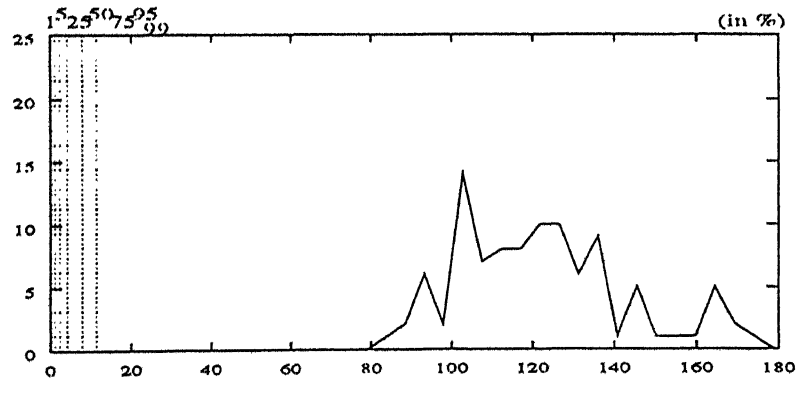

Figure 7: Frequency Test (for LCS). Block size =2 bits.

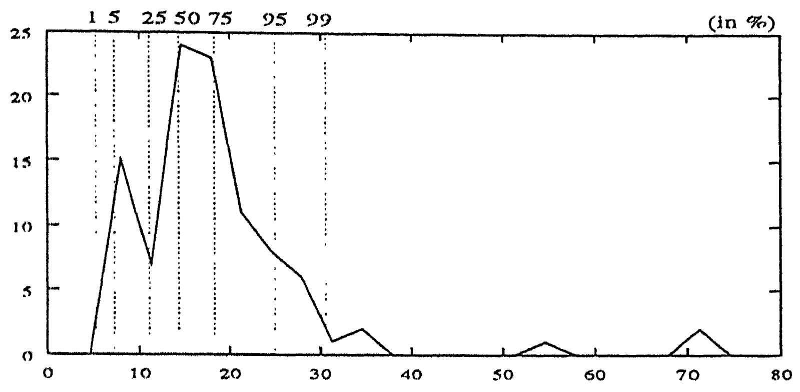

Figure 8: Frequency Test (for DES), Block size =4 bits.

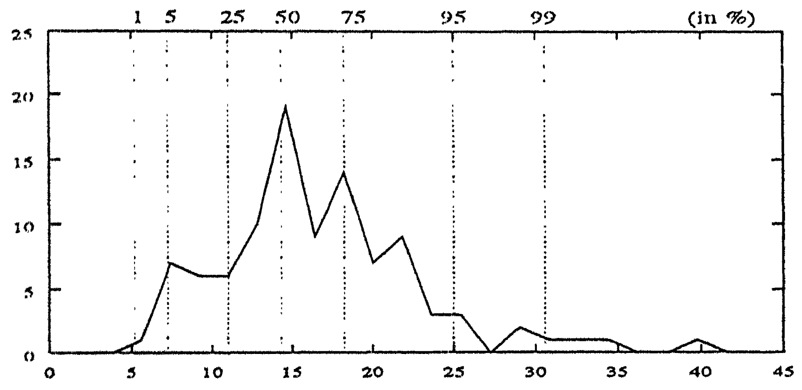

Figure 9: Frequency Test (for NCS). Block size =4 bits.

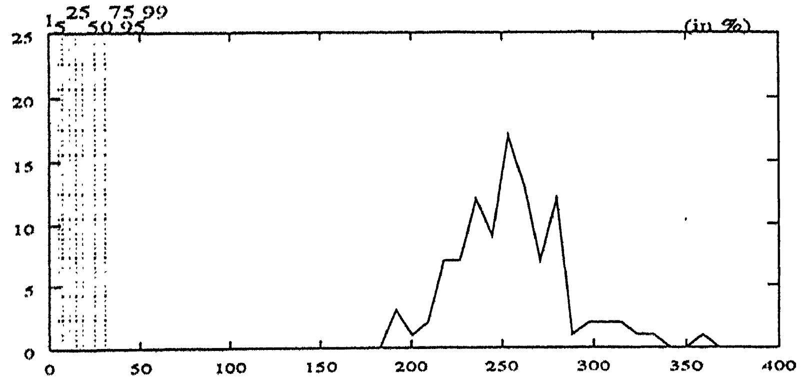

Figure 10: Frequency Test (for LCS). Block size =4 bits.

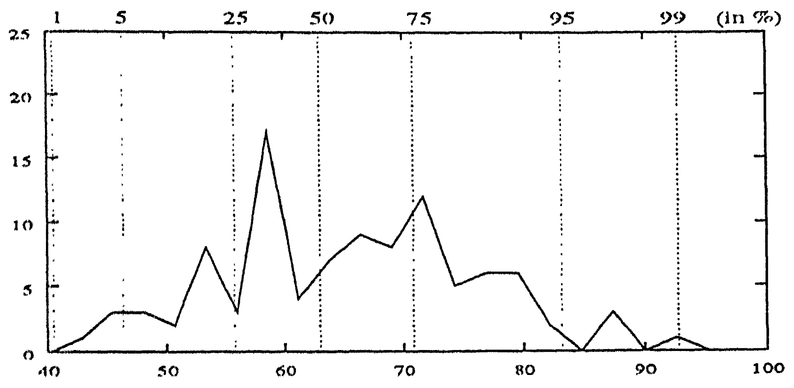

Figure 11: Serial Test (for DES), Block size =3 bits.

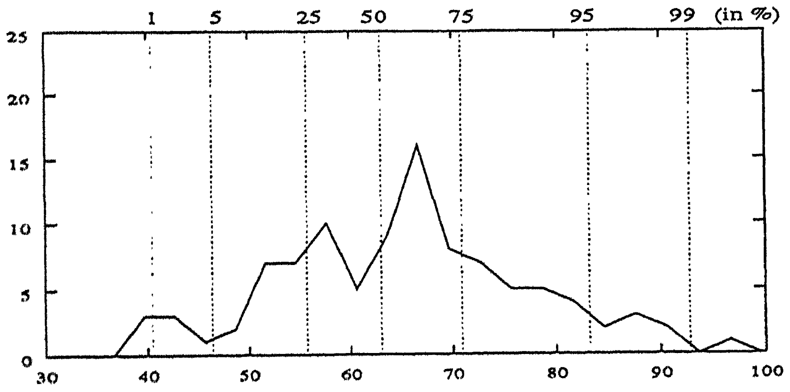

Figure 12: Serial Test (for NCS), Block size =3 bits.

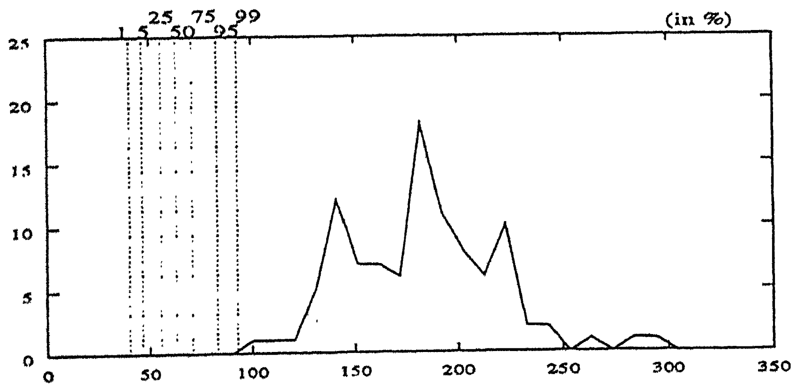

Figure 13: Serial Test (for LCS), Block size =3 bits.

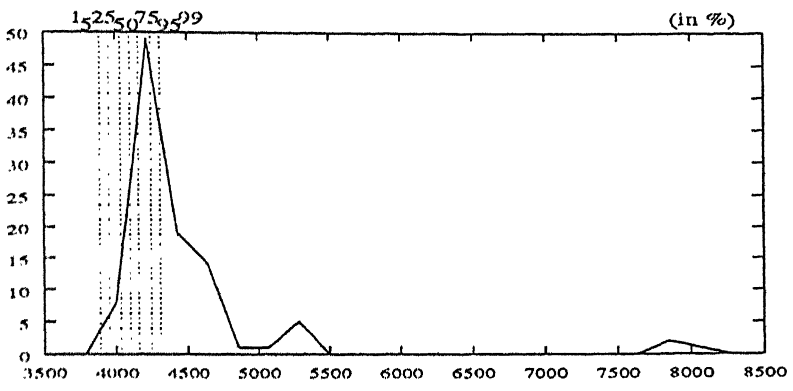

Figure 14: Serial Test (for DES), Block size =6 bits.

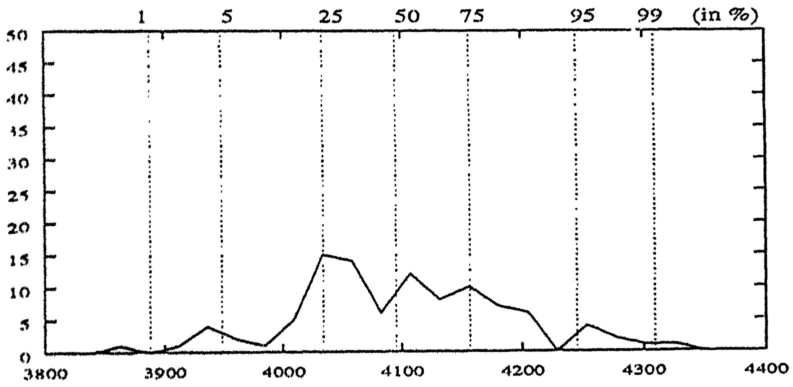

Figure 15: Serial Test (for NCS), Block size =6 bits.

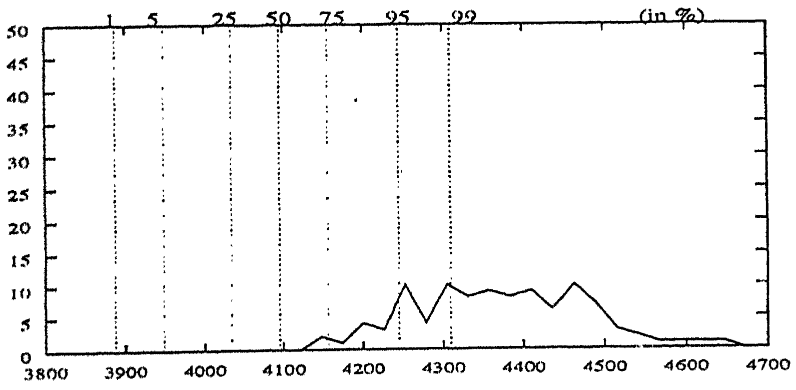

Figure 16: Serial Test (for LCS), Block size =6 bits.

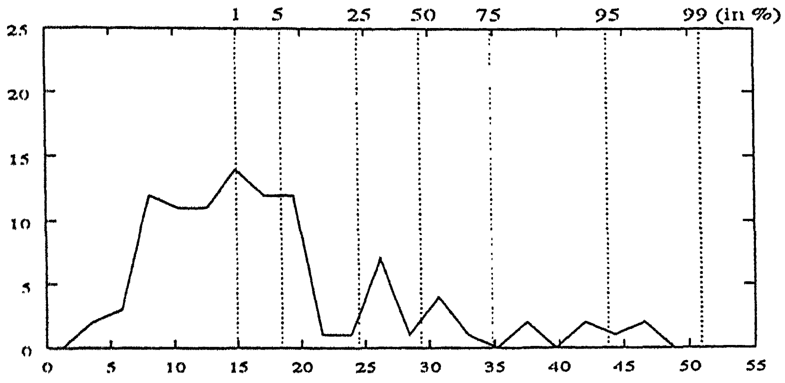

Figure 17: Gap Test (for DES), Block size =7 bits.

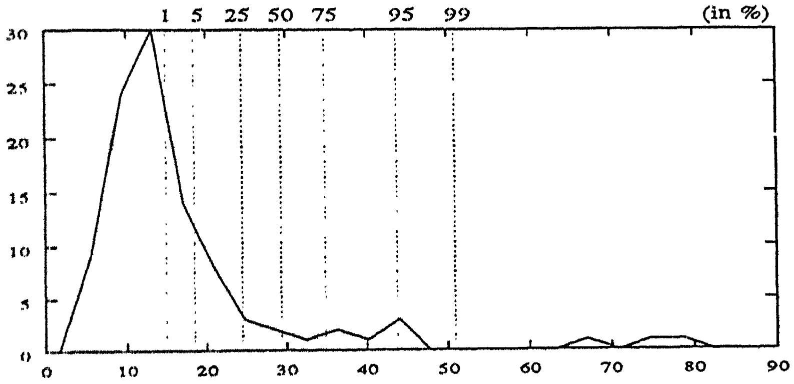

Figure 18: Gap Test (for NCS), Block size =7 bits.

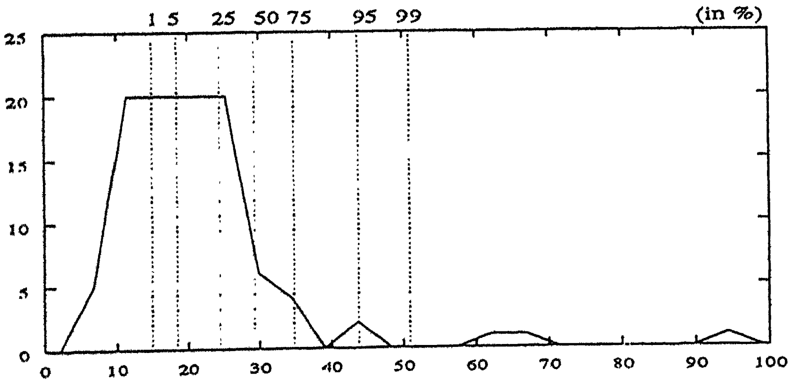

Figure 19: Gap Test (for LCS), Block size =7 bits.

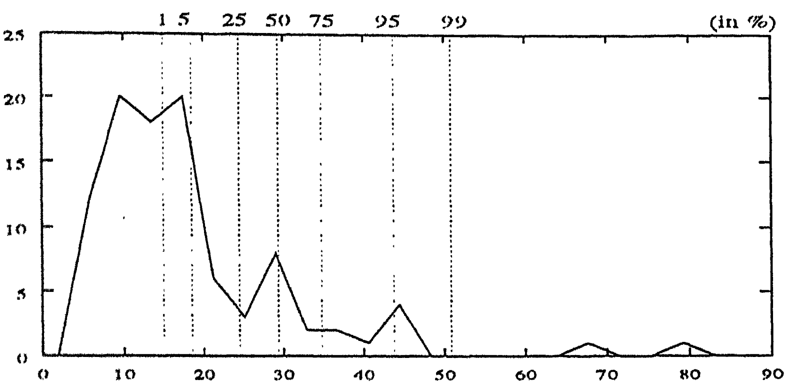

Figure 20: Gap Test (for DES), Block size =9 bits.

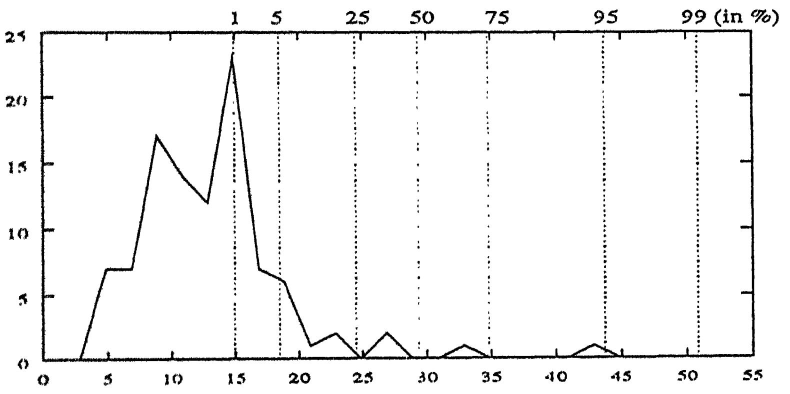

Figure 21: Gap Test (for NCS), Block size =9 bits.

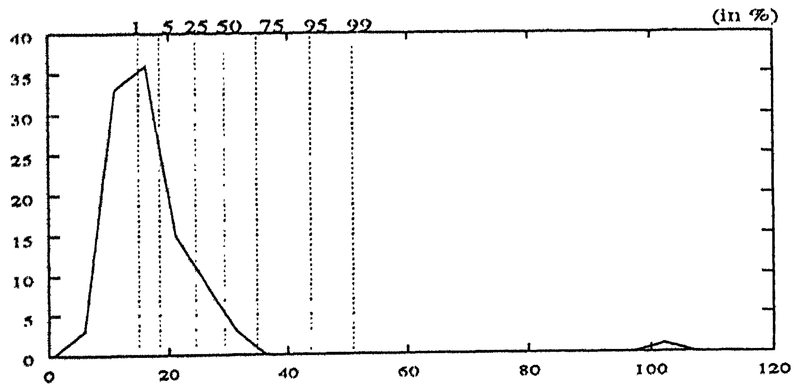

Figure 22: Gap Test (for LCS), Block size =9 bits.

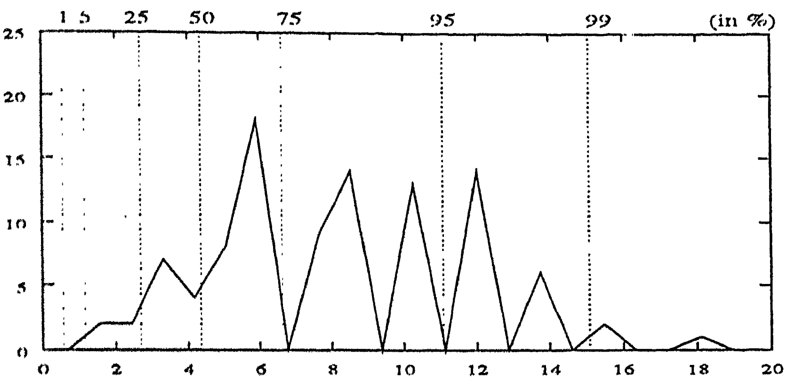

Figure 23: Poker Test (for DES), Block size =9 bits.

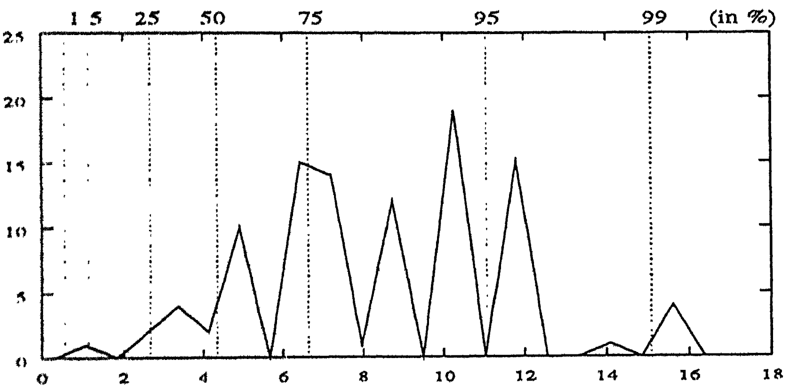

Figure 24: Poker Test (for NCS), Block size =9 bits.

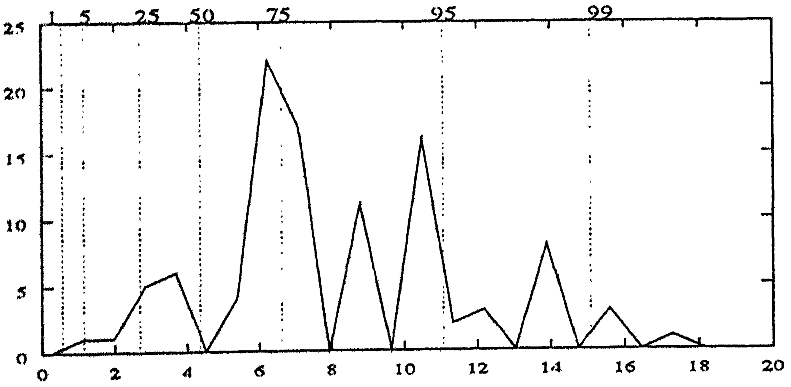

Figure 25: Poker Test (for LCS), Block size =9 bits.

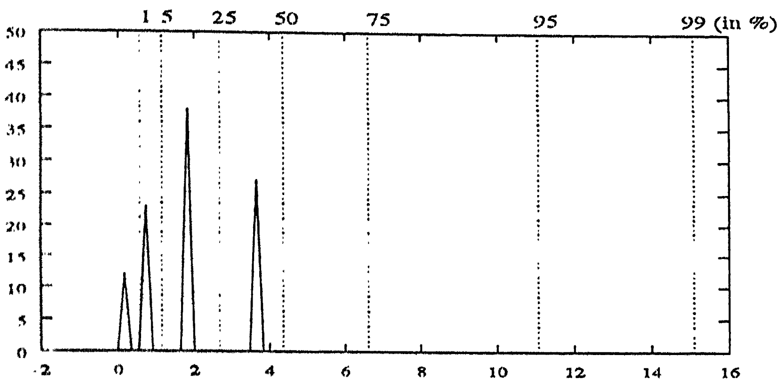

Figure 26: Poker Test (for DES), Block size =11 bits.

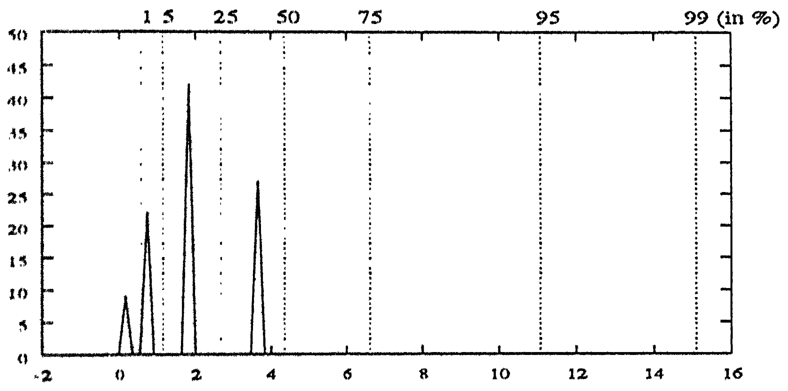

Figure 27: Poker Test (for NCS), Block size =11 bits.

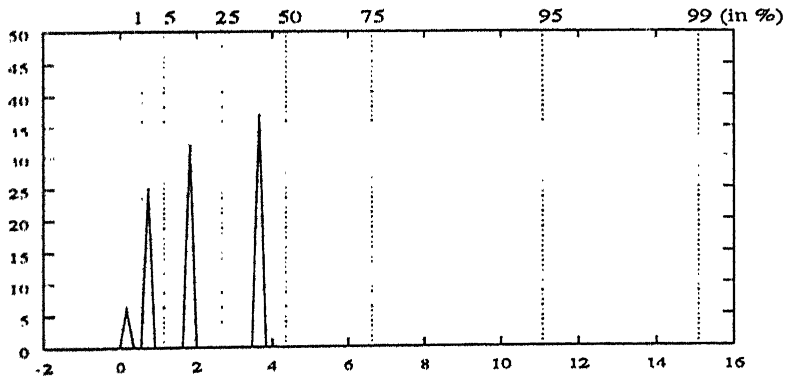

Figure 28: Poker Test (for LCS), Block size =11 bits.

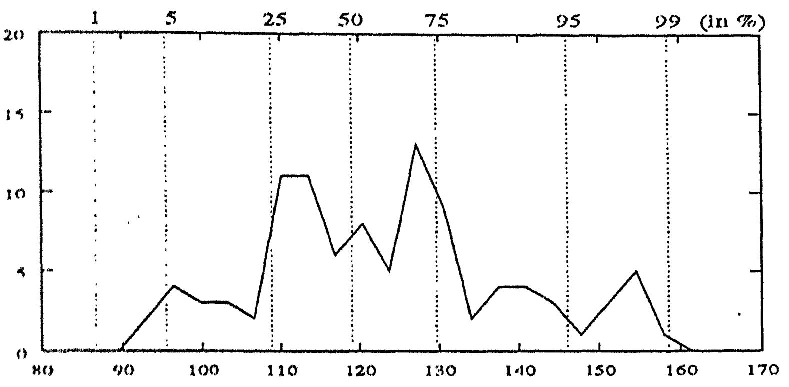

Figure 29: Permutation Test (for DES), Block size =7 bits.

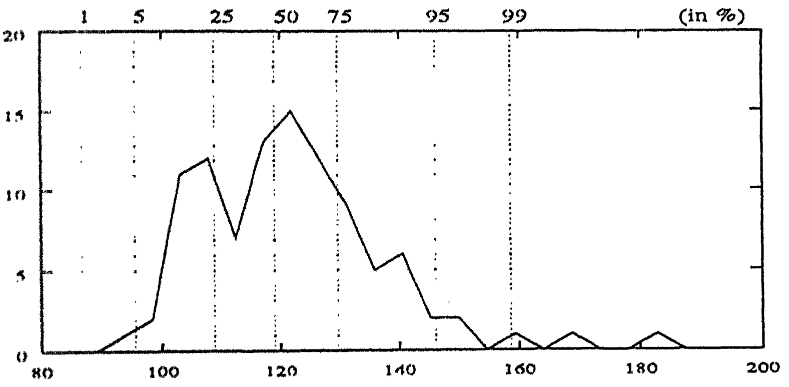

Figure 30: Permutation Test (for NCS), Block size =7 bits.

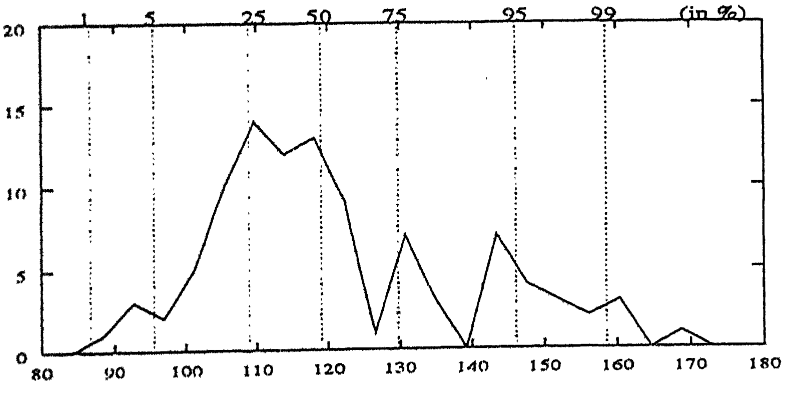

Figure 31: Permutation Test (for LCS), Block size =7 bits.

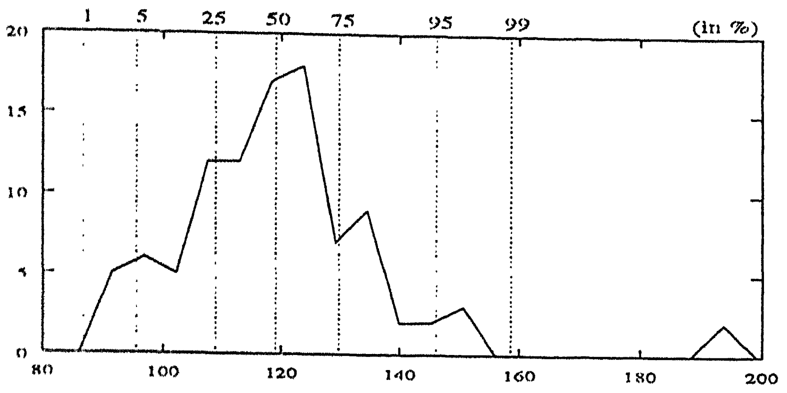

Figure 32: Permutation Test (for DES), Block size =10 bits.

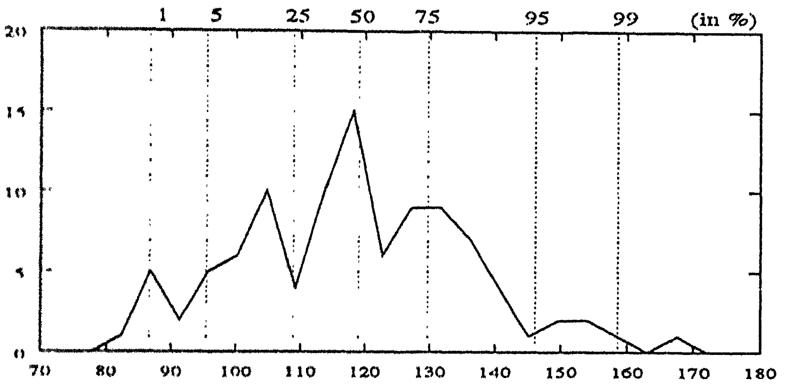

Figure 33: Permutation Test (for NCS), Block size =10 bits.

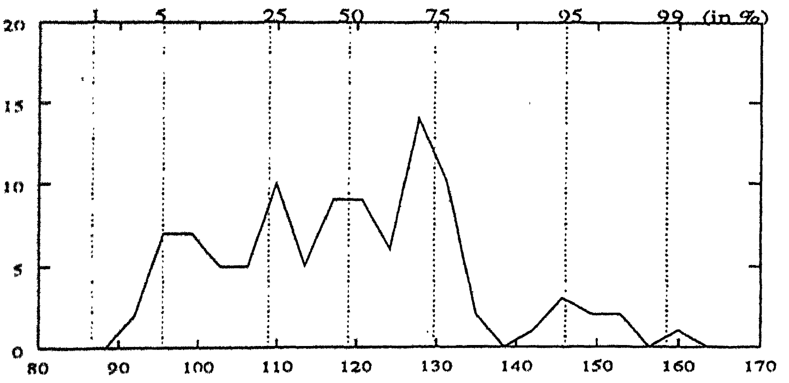

Figure 34: Permutation Test (for LCS), Block size =10 bits.

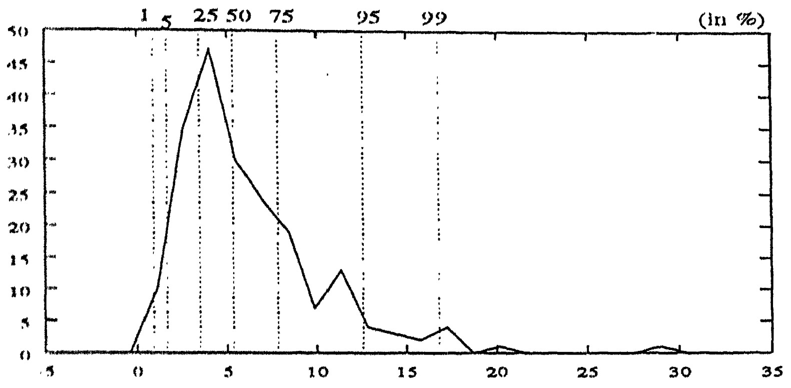

Figure 35: Run Test (for DES), Block size =6 bits.

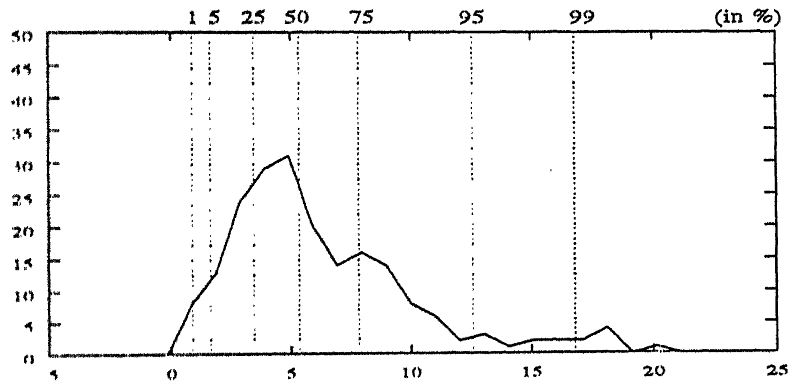

Figure 36: Run Test (for NCS), Block size =6 bits.

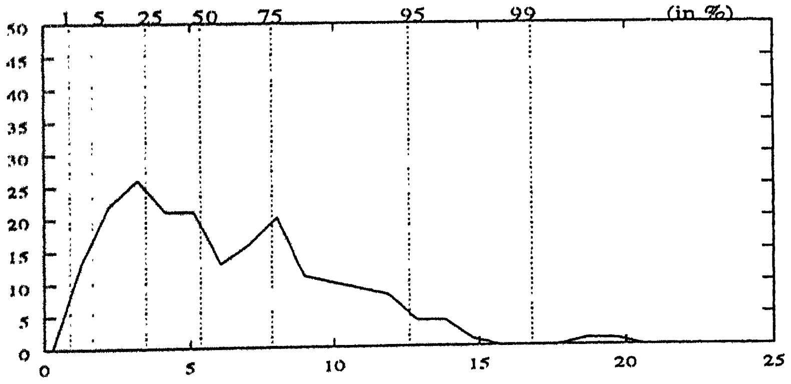

Figure 37: Run Test (for LCS), Block size =6 bits.

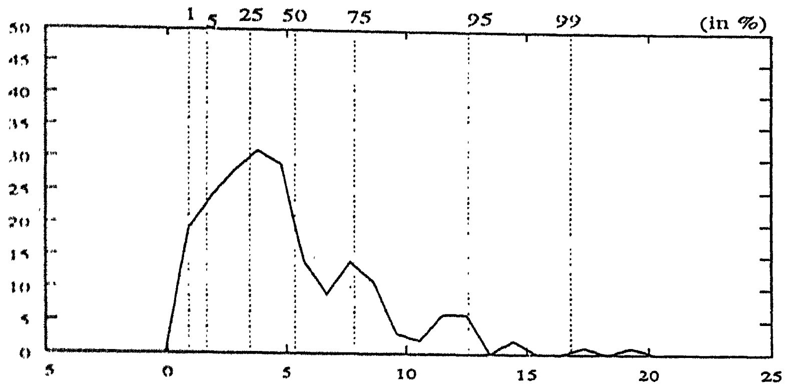

Figure 38: Run Test (for DES), Block size =12 bits.

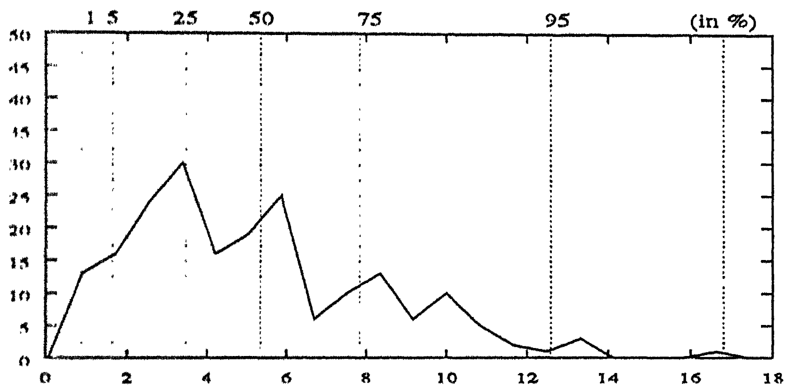

Figure 39: Run Test (for NCS), Block size =12 bits.

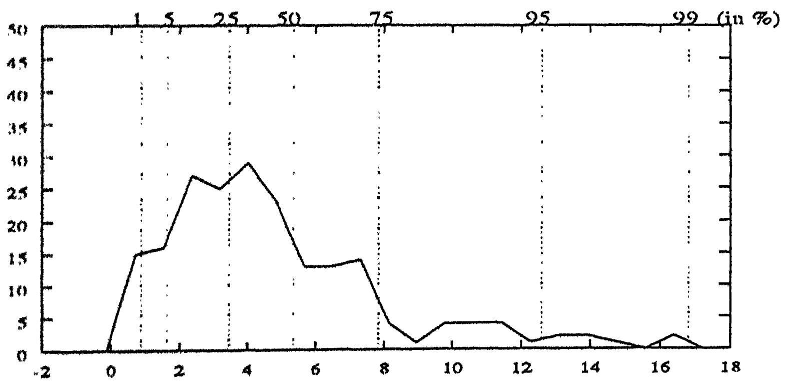

Figure 40: Run Test (for LCS), Block size =12 bits.

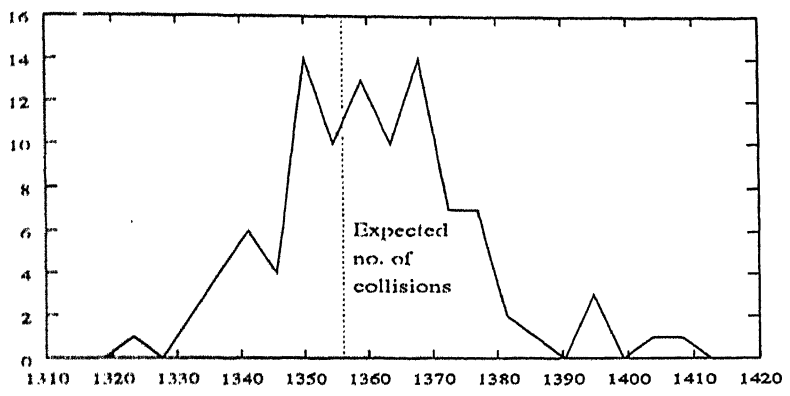

Figure 41: Collision Test (for DES), Block size = 11 bits.

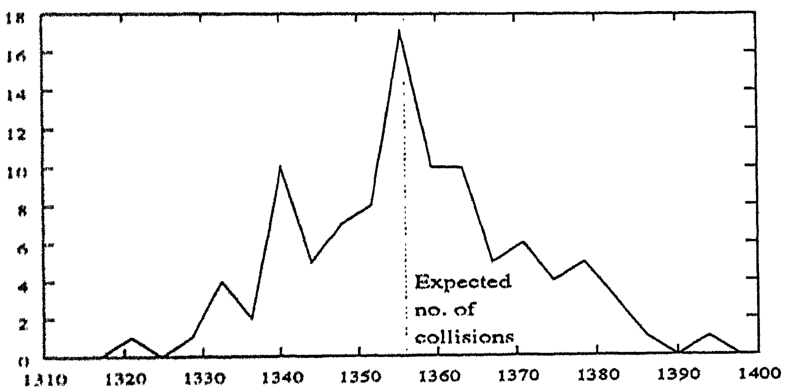

Figure 42: Collision Test (for NCS), Block size = 11 bits.

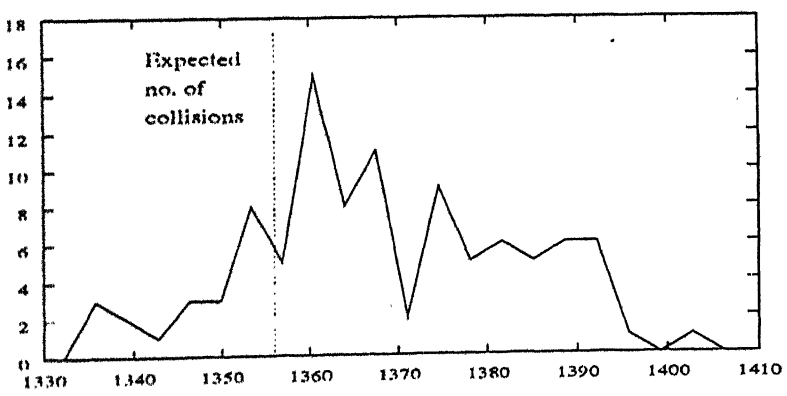

Figure 43: Collision Test (for LCS), Block size = 11 bits.

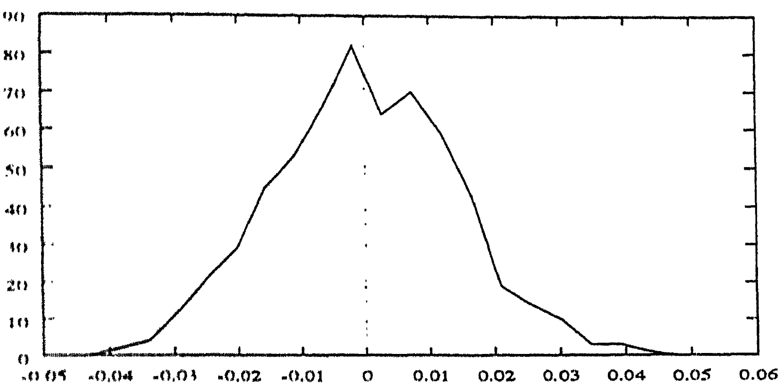

Figure 44: Correlation Test (for DES), Block size = 6 bits.

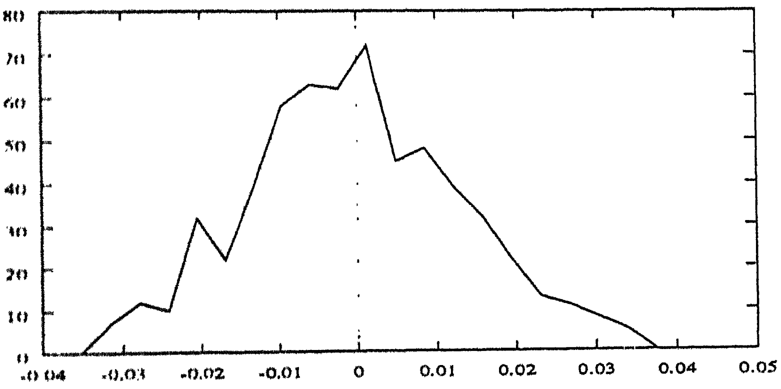

Figure 45: Correlation Test (for NCS), Block size = 6 bits.

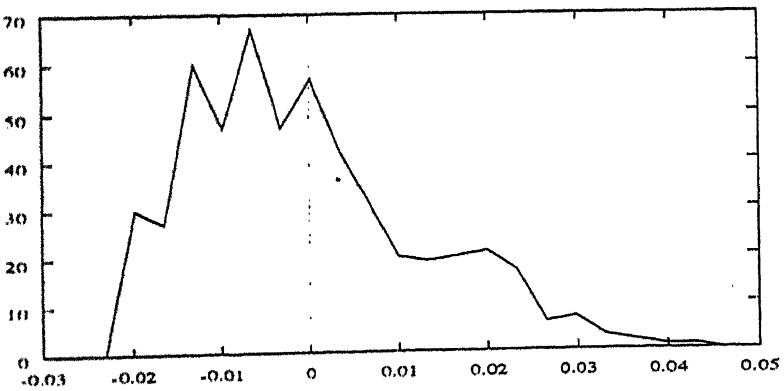

Figure 46: Correlation Test (for NCS), Block size = 6 bits.

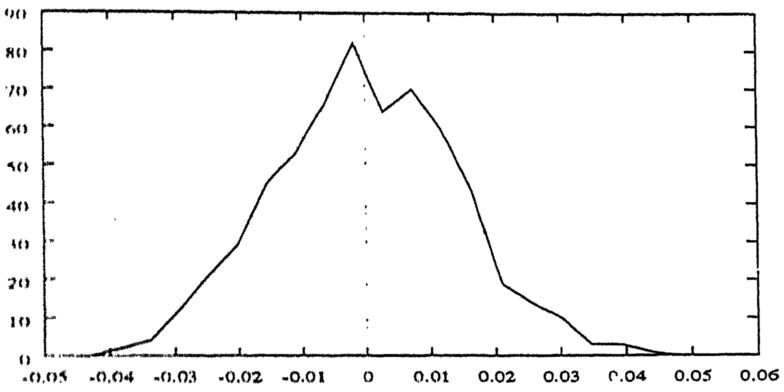

Figure 47: Correlation Test (for DES), Block size = 12 bits.

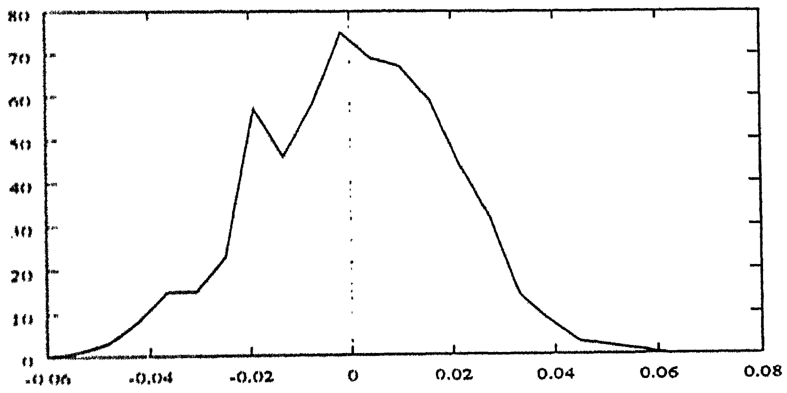

Figure 48: Correlation Test (for NCS), Block size = 12 bits.

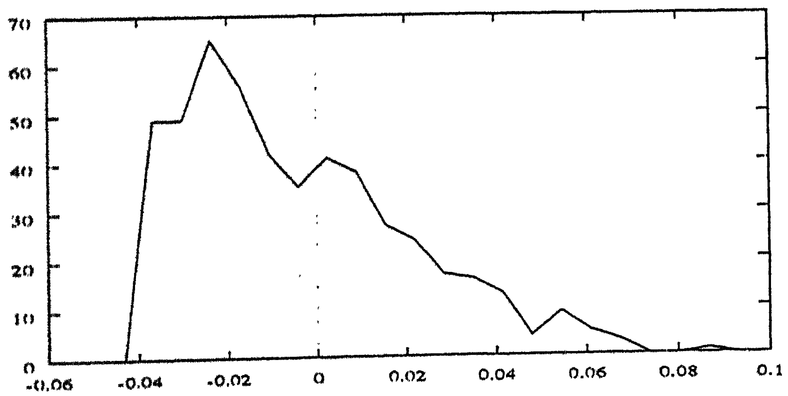

Figure 49: Correlation Test (for LCS), Block size = 12 bits.

For frequency test, block size 2 bits (shown in figs. 5 and 6), most of the chi-square values lie between 5% and 50% for DES. For NCS, many of the values are exactly equal to 50% and then slowly decrease, but for LCS (fig. 7) results are far from the range (1% to 99%) and show totally suspectable results. Similarly, for block block size 4 (figs. 8 and 9), ciphertext of NCS exhibits more random behaviour as compared to the ciphertext of DES, but all values for LCS (fig. 10) are out of range.

For serial test, block size 3 bits (figs. 11 and 12), values are dispersed around 50% line for DES and NCS and it shows the same random behaviour. But again all of the values are out of range for LCS (fig. 13). For block size 6 bits (fig. 14) most of the values for DES lie between 50% and 75%, and there are some values which are far from random behaviour, but almost all of the values for NCS (fig. 15) show random behaviour and almost equally distributed between 25% and 75% (better than DES). For LCS (fig. 16), more than 50% values still lie out of range.

None of the results are satisfactory to show the randomness of the ciphertext with respect to gap test, because more than 50% results do not even fall within the range, and this is true for all DES, LCS and NCS (figs. 17, 18, 19, 20, 21 and 22).

For poker test, all the three cryptosystems show the same random behaviour. Although, very few observations fall out of range, but the distribution of numbers show suspectable behaviour of ciphertexts(figs. 23, 24, 25, 26, 27 and 28).

Permutation test (figs. 29 and 30) clearly show that the ciphertext of NCS is more random than the ciphertext of DES for block size 7 bits, because the peak of the curve almost lie at 50% and then decrease slowly. In the case of DES, the values are distributed between 25% and 75%. Results of LCS (fig. 31) appear better than the results of DES, but NCS is producing the best results. In case of block size 10 bits (figs. 32, 33 and 34), random behaviour of DES and NCS is not differing that much and is better than that of LCS.

For run test, results of LCS are little better than the results of DES and NCS for block size 6 bits(figs. 35, 36 and 37). But for block size 12 (figs. 38, 39 and 40) results display almost equal randomness for all of these cryptosystems.

Collision test curves show that almost 50% observations give more number of collisions than the expected number of collisions in case of DES and NCS (figs. 41 and 42), but for LCS (fig. 43) number of collision is more than the expected number of collisions for more than 75% observations.

As described in chapter 3, the correlation cofficients should be very near to 0 to show linear dependency. And both of the ciphertext show random behaviour (figs. 44, 45, 47 and 48). The LCS curve is little shifted to negative side, but still these show the random behaviour of ciphertext(fig. 46 and 49).

These randomness tests are performed for two different size of blocks, though the random behaviour of ciphertext for one block size cannot guarantee the random behaviour for any other block size, but still these graphs give us some assurance about the randomness of the ciphertext.

These examples show that, in most of the tests, the NCS is producing more or equal random output as compared to DES. But LCS is not producing satifactory results in most of the cases. In short, we can say that new cryptosystem for small round and small key size is producing better results, however, for more rounds and larger key size the randomness is getting reduced! The reasons for this seemingly counterintutive behaviour need further study. We make some guesses as to why this is occuring:

(i) The key used for NCS is in some way a "better" key than that one used for LCS. Perhaps a different key for LCS would also yield good random behaviour.

(ii) Every round of the new cryptosystem increases the size of the text by between 0.8% to 1.5%. So for 10 rounds, the increase in size may be almost 8%. This may also reflect on the randomness.

On the average, we can say that ciphertext produced by new cryptosystem is atleast as "good" as ciphertext by DES for small number of rounds and small key size.

4.2 Conclusion

As for DES and NCS, it can be safely said that NCS produced more random output than DES. From the discussion above, it is clear that the DES is very sensitive to differential cryptanalysis and easy to break for small number of rounds, but the added properties in the new cryptosystem make it immune to differential cryptanalysis. The ciphertext produced by the new cryptosystem for small number of rounds is more random than the ciphertext produced by DES, and probably very difficult to analyse by other cryptanalysis schemes too.

4.3 Future Directions

In this thesis work, a new cryptosystem is analysed against differential cryptanalysis. Though random output and a study of added properties show that it is very strong cryptosystem, but it should also be analysed against other cryptanalysis schemes. Here we implemented automated password generator using DES (chapter 5), but it can also be implemented using new cryptosystem, as the new cryptosystem generates more random text as output.

Chapter 5

Automated Password Generator (APG) using DES

A password is a string of characters used to authenticate the identity of a computer system user, or to authorize access to system resources, but when users are allowed to choose their password they often select passwords related to their personal information, like, names of children, pets, favorite foods, favorite places etc. These types of passwords can be predicted by others, those having this information about the user. So the user should select a password which he can remember but others can not guess.

An automated password generator (APG) creates random password that is not related to the user. In this algorithm, random numbers are used to select the characters that form random yet pronounceable passwords, and these random numbers are generated by a random number subroutine based on DES. The old password is input to DES, and a pseudorandom key is generated with the help of Unix system's command (provided to generate pseudorandom numbers). DES generates an entirely different random number, that is used to create random password.

5.1 Implementation

There are three main components which are referred by the main procedure in order to implement APG. These are explained below.

5.1.1 Unit Table

The unit table defines the units and specifies rules pertaining to the individual units used in randomly generated word, so that the word is pronounceable. These units are combined to form pronounceable syllables, and by concatenating them a word can be formed. These unit rules determine whether a given unit is legal or illegal, based on its position within the syllable and adjacent units. These rules are not dependent on anything outside the current syllable.

5.1.2 Digram Table

This table specifies the rules about all possible pairs of units and the rules to juxtapose these units. The table contains one entry for every pair of units, giving rules whether that pair is allowed or not, and in which position in the syllable it is allowed and in which position it is not allowed. The random word generater generates the pronounceable word with the help of these rules.

5.1.3 Random Number Generator Subroutine

The random number generator uses a subroutine in which random number is generated with the help of DES. The plaintext (64 bit block) entered to the DES will be created by the first eight bytes of old password or a data string, which are entered manually from the keyboard. if the old password or data string is less than eight characters long the extra elements in the input are filled with ASCII 0. The 64 bit key is generated using Unix system's pseudorandom password generator and seed value is generated by using the current time at the time of execution. DES is then used to encrypt the input data, the output is eight byte character string(64 bit block), say represented by character array *out*. To obtain a random number from this character string, the ASCII values of the first three elements are added and stored in variable *sum*. Thus

$$sum = out[0] + out[1] + out[2]$$

and *sum* is an integer. To obtain a random number within the required range of 0 to $n-1$ from *sum*, the function takes the modulus of *sum* and n,($sum\%n$). This value is returned to calling function within the random number program.

5.2 Random Word Algorithm

The function of the random word generator is to determine whether a given unit, generated by random unit subroutine, can be appended to the end of the partial word formed so far. The random unit subroutine generates random numbers using the DES function. This function is called many times. It asks for the old password the first time and then sends that password to the DES function on every successive call afterwards and in every call a different random key is used.

Rules of pronounceability are stored in the unit and digram tables. These rules check whether a unit is legal or illegal. If illegal, the unit is discarded and the random unit subroutine is called again. If unit is accepted, the state variables are updated and we try for the next unit of the word. Most of the rules are syllable oriented and do not depend on anything outside the current syllables. When the end of the word is reached, some additional checks are made before termination of algorithm.

A block diagram for APG algorithm is shown in fig. 50.

The DES randomizer accepts an old password and a pseudorandom key (generated with the help of unix system command) and generates a random number. This number is used by Random Word Generator to develop a password. As the password is

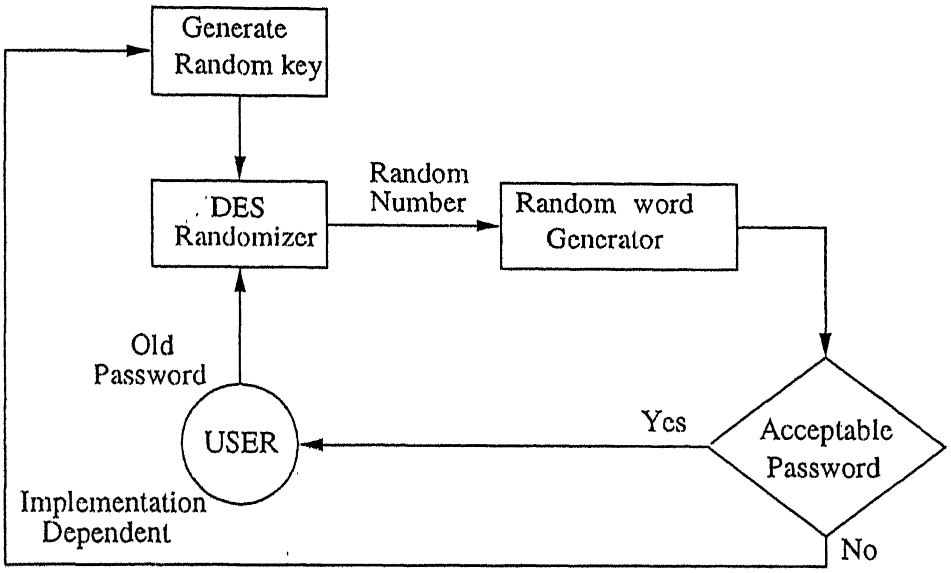

Figure 50: Block Diagram of APG

being generated, each group of letters is subjected to tests of grammar and semantics to determine if this is acceptable word. If all tests are passed, a new password is given to the user.

5.3 Applications of APG

APG produces passwords those are not associated with the user. As these are pronounceable, so they are easy to remember and enter in a computer system, yet these are not susceptible to the automated techniques used to break the passwords.

APG uses only 26 characters of English alphabet and doesn't use any number or special character, but still its key space is very large. It is approximately 18 millions for 6 character password and 1.6 trillion for 10 character password.

So APG is very efficient scheme to generate secure passwords.

References

[1] D. E. Knuth. *The Art of Computer Programming. Volume 2: Seminumerical Algorithms.* Addison-Wesley Series in Computer Science and Information Processing.

[2] B. Schneier. *Applied Cryptography.* John Wiley & Sons, 1994.

[3] E. Biham and A. Shamir. Differential Cryptanalysis of DES-like Cryptosystems. *Advances in Cryptology - CRYPTO '90 proceedings,* Berlin: Springer-Verlag, 1991, pp. 2-21.

[4] E. Biham and A. Shamir. Differential Cryptanalysis of FEAL and N-HASH. *Advances in Cryptology - EUROCRYPT '91 proceedings,* Berlin: Springer-Verlag, 1991, pp. 1-16.

[5] I. Ben-Aroya and E. Biham. Differential Cryptanalysis of Lucifer. *Advances in Cryptology - CRYPTO '93,* Berlin: Springer-Verlag, 1991, pp. 2-21.

[6] Thomas A. Berson. Differential Mod 2^{32} with Applications to MD5. *Advances in Cryptology - EUROCRYPT '92,* Berlin: Springer-Verlag, 1992, pp. 71-80.

[7] W. Meier and O. Staltelbach. Non-linearity Criteria for Cryptographic Functions. *Advances in Cryptology - EUROCRYPT '89,* Berlin: Springer-Verlag, 1989, pp. 549-562.

[8] Automated password Generator (APG) *Federal Information Processing Standards Publication (FIPS PUB) 181* National Institute of Standards and Technology, 1993, pp. 1-53.